Scoup-SMT: Scalable Coupled Sparse Matrix-Tensor Factorization

Scoup-SMT: Scalable Coupled Sparse Matrix-Tensor Factorization

Evangelos E. Papalexakis
Carnegie Mellon University
epapalex@cs.cmu.edu

Tom M. Mitchell
Carnegie Mellon University
tom.mitchell@cmu.edu

Nicholas D. Sidiropoulos
University of Minnesota
nikos@ece.umn.edu

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

Partha Pratim Talukdar
Carnegie Mellon University
partha.talukdar@cs.cmu.edu

Brian Murphy
Carnegie Mellon University
brianmurphy@cmu.edu

ABSTRACT

How can we correlate neural activity in the human brain as it responds to words, with behavioral data expressed as answers to questions about these same words? In short, we want to find latent variables, that explain both the brain activity, as well as the behavioral responses. We show that this is an instance of the *Coupled Matrix-Tensor Factorization* (CMTF) problem. We propose SCOUP-SMT, a novel, fast, and parallel algorithm that solves the CMTF problem and produces a *sparse* latent low-rank subspace of the data. In our experiments, we find that SCOUP-SMT is *50-100 times* faster than a state-of-the-art algorithm for CMTF, along with a *5 fold* increase in sparsity. Moreover, we extend SCOUP-SMT to handle missing data without degradation of performance.

We apply SCOUP-SMT to BRAINQ, a dataset consisting of a (nouns, brain voxels, human subjects) tensor and a (nouns, properties) matrix, with coupling along the nouns dimension. SCOUP-SMT is able to find meaningful latent variables, as well as to predict brain activity with competitive accuracy. Finally, we demonstrate the generality of SCOUP-SMT, by applying it on a FACEBOOK dataset (users, 'friends', wall-postings); there, SCOUP-SMT spots spammer-like anomalies.

Keywords

Tensor Decompositions, Coupled Matrix-Tensor Factorization, Sparsity, Parallel Algorithm, Brain Activity Analysis

1. INTRODUCTION

How is knowledge mapped and stored in the human brain? How is it expressed by people answering simple questions about specific words? If we have data from both worlds, are we able to combine them and jointly analyze them? In a very different scenario, suppose we have the social network graph of an online social network, and we also have additional information about how and when users interacted with each other. What is a comprehensive way to combine those two pieces of data? Both, seemingly different, problems may be viewed as instances of what is called *Coupled Matrix-Tensor Factorization* (CMTF), where a data tensor and matrices that hold additional information are jointly decomposed into a set of low-rank factors.

In this work, we introduce SCOUP-SMT, a fast, scalable, and sparsity promoting CMTF algorithm. Our main contributions are the following:

- *Fast, parallel & sparsity promoting algorithm:* We provide a novel, scalable, and sparsity inducing algorithm, SCOUP-SMT, that jointly decomposes coupled matrix-tensor data. Figure 1 shows the accuracy of SCOUP-SMT (compared to the traditional algorithm), as a function of portion of the wall-clock time that our algorithm took, again compared to the traditional one. The result indicates a speedup of about 50-100 times, while maintaining very good accuracy.[1]
- *Robustness to missing data:* We carefully derive an improved version of the above algorithm which is resilient to missing data and performs well, even with a large portion of the entries missing.
- *Effectiveness & Knowledge Discovery:* We analayze BRAINQ, a brain scan dataset which is coupled to a semantic matrix (see Sec. 4 for details). The brain scan part of the dataset consists of fMRI scans first used in [16], a work that first demonstrated that brain activity can be predictably analyzed into component semantic features. Here, we demonstrate a disciplined way to combine both datasets and carry out a variety of data mining/machine learning tasks, through this joint analysis.
- *Generality*: We illustrate the generality of our approach, by applying SCOUP-SMT to a completely different setting of a time-evolving social network with side information on user interactions, demonstrating SCOUP-SMT's ability to discover anomalies.

2. PRELIMINARIES

2.1 Introduction to Tensors

Matrices record dyadic properties, like "people recommending products". Tensors are the n-mode generalizations, capturing 3- and higher-way relationships. For example "subject-verb-object" relationships, such as the ones recorded by the Read the Web - NELL project [1] (and have been recently used in this context [10]

[1]Accuracy or relative cost is defined in Section 5 as the ratio of the squared approximation error of SCOUP-SMT, divided by that of the traditional ALS algorithm.

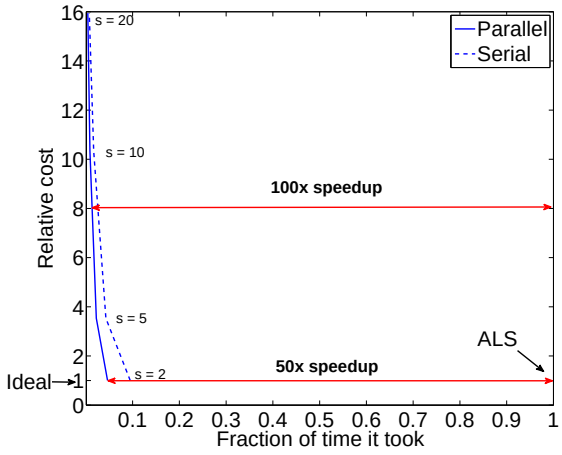


Figure 1: The relative cost of SCOUP-SMT (with respect to the ALS algorithm) as a function of the fraction of the wall-clock time of ALS that the computation required, vividly demonstrates the gains of SCOUP-SMT in terms of speedup. In particular, for the entire BRAINQ dataset which is very dense (see Sec. 4), the speedup incurred by the parallel version of SCOUP-SMT on 4 cores, was in the range of 50-100 times. This Figure also shows the behavior of SCOUP-SMT with respect to the sampling parameter s. As s increases, SCOUP-SMT runs faster but the relative cost increases as well.

Symbol	Description
CMTF	Coupled Matrix-Tensor Factorization
ALS	Alternating Least Squares
$x, \mathbf{x}, \mathbf{X}, \underline{\mathbf{X}}$	scalar, vector, matrix, tensor (respectively)
$\mathbf{A} \odot \mathbf{B}$	Khatri-rao product (see [12]).
$\mathbf{A} * \mathbf{B}$	Hadamard (elementwise) product.
$\mathbf{A}^\dagger$	Pseudoinverse of $\mathbf{A}$ (see Sec. 2)
$\|\mathbf{A}\|_F$	Frobenius norm of $\mathbf{A}$.
$\mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$	$(\mathbf{a} \circ \mathbf{b} \circ \mathbf{c})\,(i,j,k) = \mathbf{a}(i)\mathbf{b}(j)\mathbf{c}(k)$
(i) as superscript	Indicates the i-th iteration
$\mathbf{A}_1^i, \mathbf{a}_1^i$	series of matrices or vectors, indexed by i.
$\underline{\mathbf{X}}_{(i)}$	i-th mode unfolding of tensor $\underline{\mathbf{X}}$ (see [11]).
$\mathcal{I}$	Set of indices.
$\mathbf{x}(\mathcal{I})$	Spanning indices $\mathcal{I}$ of $\mathbf{x}$.

Table 1: Table of symbols

[19]) naturally lead to a 3-mode tensor. In this work, our working example of a tensor has three modes. The first mode contains a number of nouns; the second mode corresponds to the brain activity, as recorded by an fMRI machine; and the third mode identifies the human subject corresponding to a particular brain activity measurement.

Earlier [19] we introduced a scalable and parallelizable tensor decomposition which uses mode sampling. In this work, we focus on a more general and expressive framework, that of *Coupled Matrix-Tensor Factorizations*.

2.2 Coupled Matrix-Tensor Factorization

Oftentimes, two tensors, or a matrix and a tensor, may have one mode in common; consider the example that we mentioned earlier, where we have a word by brain activity by human subject tensor, we also have a semantic matrix that provides additional information

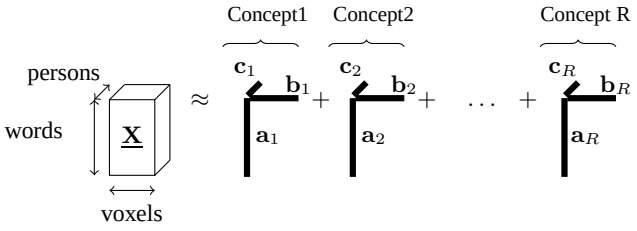


Figure 2: PARAFAC decomposition of a three-way tensor of a brain activity tensor as sum of F outer products (rank-one tensors), reminiscing of the rank-F singular value decomposition of a matrix. Each component corresponds to a **latent** concept of, e.g. "insects", "tools" and so on, a set of brain regions that are most active for that particular set of words, as well as groups of persons.

for the same set of words. In this case, we say that the matrix and the tensor are *coupled* in the 'subjects' mode.

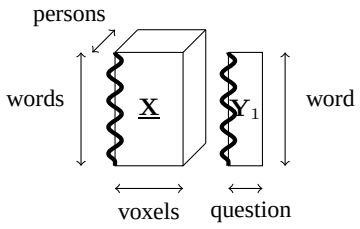


Figure 3: *Coupled Matrix - Tensor* example: Tensors often share one or more modes (with thick, wavy line): $\underline{\mathbf{X}}$ is the brain activity tensor and $\mathbf{Y}$ is the semantic matrix. As the wavy line indicates, these two datasets are coupled in the 'word' dimension.

In this work we focus on three mode tensors, however, everything we mention extends directly to higher modes. In the general case, a three mode tensor $\underline{\mathbf{X}}$ may be coupled with at most three matrices $\mathbf{Y}_i,\ i = 1 \cdots 3$, in the manner illustrated in Figure 3 for one mode. The optimization function that encodes this decomposition is:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D},\mathbf{E},\mathbf{G}} \|\underline{\mathbf{X}} - \sum_k \mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k\|_F^2 + \quad (1)$$

$$\|\mathbf{Y}_1 - \mathbf{A}\mathbf{D}^T\|_F^2 + \|\mathbf{Y}_2 - \mathbf{B}\mathbf{E}^T\|_F^2 + \|\mathbf{Y}_3 - \mathbf{C}\mathbf{G}^T\|_F^2$$

where $\mathbf{a}_k$ is the k-th column of $\mathbf{A}$. The idea behind the coupled matrix-tensor decomposition is that we seek to jointly analyze $\underline{\mathbf{X}}$ and $\mathbf{Y}_i$, decomposing them to latent factors who are coupled in the shared dimension. For instance, the first mode of $\underline{\mathbf{X}}$ shares the same low rank column subspace as $\mathbf{Y}_1$; this is expressed through the latent factor matrix $\mathbf{A}$ which jointly provides a basis for that subspace.

2.3 The Alternating Least Squares Algorithm

One of the most popular algorithms to solve PARAFAC (as introduced in Figure 2) is the so-called Alternating Least Squares (ALS); the basic idea is that by fixing two of the three factor matrices, we have a least squares problem for the third, and we thus do so iteratively, alternating between the matrices we fix and the one we optimize for, until the algorithm converges, usually when the relative change in the objective function between two iterations is very small.

Solving CMTF using ALS follows the same strategy, only now, we have up to three additional matrices in our objective. For instance, when fixing all matrices but $\mathbf{A}$, the update for $\mathbf{A}$ requires to solve the following least squares problem:

$$\min_{\mathbf{A}} \|\underline{\mathbf{X}}_{(1)} - (\mathbf{B} \odot \mathbf{C})\mathbf{A}^T\|_F^2 + \|\mathbf{Y}_1 - \mathbf{D}\mathbf{A}^T\|_F^2$$

In Algorithm 1, we provide a detailed outline of the ALS algorithm for CMTF.

In order to obtain initial estimates for matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ we take the PARAFAC decomposition of $\underline{\mathbf{X}}$. As for matrix $\mathbf{D}$ (and similarly for the rest), it suffices to solve a simple Least Squares problem, given the PARAFAC estimate of $\mathbf{A}$, we initialize as $\mathbf{D} = \mathbf{Y}_1 \left(\mathbf{A}^\dagger\right)^T$, where $\dagger$ denotes the Moore-Penrose Pseudoinverse which, given the Singular Value Decomposition of a matrix $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, is computed as $\mathbf{X}^\dagger = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T$.

Algorithm 1: Alternating Least Squares Algorithm for CMTF

Input: $\underline{\mathbf{X}}$ of size $I \times J \times K$, matrices $\mathbf{Y}_i$, $i = 1 \cdots 3$, of size $I \times I_2$, $J \times J_2$, and $K \times K_2$ respectively, number of factors F.

Output: $\mathbf{A}$ of size $I \times F$, $\mathbf{b}$ of size $J \times F$, $\mathbf{c}$ of size $K \times F$, $\mathbf{D}$ of size $I_2 \times F$, $\mathbf{G}$ of size $J_2 \times F$, $\mathbf{E}$ of size $K_2 \times F$.

1: Unfold $\underline{\mathbf{X}}$ into $\underline{\mathbf{X}}_{(1)}, \underline{\mathbf{X}}_{(2)}, \underline{\mathbf{X}}_{(3)}$ (see [11]).

2: Initialize $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ using PARAFAC of $\underline{\mathbf{X}}$. Initialize $\mathbf{D}, \mathbf{G}, \mathbf{E}$ as discussed on the text.

3: **while** convergence criterion is not met **do**

4: $\mathbf{A} = \begin{bmatrix} \underline{\mathbf{X}}_{(1)} \\ \mathbf{Y}_1 \end{bmatrix}^T \left(\begin{bmatrix} (\mathbf{B} \odot \mathbf{C}) \\ \mathbf{D} \end{bmatrix}^\dagger \right)^T$

5: $\mathbf{B} = \begin{bmatrix} \underline{\mathbf{X}}_{(2)} \\ \mathbf{Y}_2 \end{bmatrix}^T \left(\begin{bmatrix} (\mathbf{C} \odot \mathbf{A}) \\ \mathbf{G} \end{bmatrix}^\dagger \right)^T$

6: $\mathbf{C} = \begin{bmatrix} \underline{\mathbf{X}}_{(3)} \\ \mathbf{Y}_3 \end{bmatrix}^T \left(\begin{bmatrix} (\mathbf{A} \odot \mathbf{B}) \\ \mathbf{E} \end{bmatrix}^\dagger \right)^T$

7: $\mathbf{D} = \mathbf{Y}_1 \left(\mathbf{A}^\dagger\right)^T,\ \mathbf{G} = \mathbf{Y}_2 \left(\mathbf{B}^\dagger\right)^T,\ \mathbf{E} = \mathbf{Y}_3 \left(\mathbf{C}^\dagger\right)^T$

8: **end while**

Besides ALS, there exist other algorithms for CMTF. For example, [4] uses a first order optimization algorithm for the same objective. However, we chose to operate using ALS because it is the 'workhorse' algorithm for plain tensor decomposition, and it easily to incorporate additional constraints in ALS. Nevertheless, one strength of SCOUP-SMT is that it can be used as-is with any underlying core CMTF implementation.

3. PROPOSED METHOD

3.1 Algorithm description

There are three main concepts behind SCOUP-SMT (outlined in Algorithm 2):

Phase 1 Sample the data in order to reduce the dimensionality

Phase 2 fit CMTF to the reduced data (possibly on more than one samples)

Phase 3 merge the partial results

Phase1: Sampling An efficient way to reduce the size of the dataset, yet operate on a representative subset thereof is to use *biased* sampling. In particular, given a three-mode tensor $\underline{\mathbf{X}}$ we sample as follows. We calculate three vectors as shown in equation (2), one for each mode of $\underline{\mathbf{X}}$. These vectors, which we henceforth refer to as *density vectors* are the marginal absolute sums with respect to all but one of the modes of the tensor, and in essence represent the importance of each index of the respective mode. We then sample *indices* of each mode according to the respective density vector. For instance, assume an $I \times J \times K$ tensor; suppose that we need a sample of size $\frac{I}{s}$ of the indices of the first mode. Then, we just define

$$p_{\mathcal{I}}(i) = \mathbf{x}_A(i) / \sum_{i=1}^{I} \mathbf{x}_A(i)$$

as the probability of sampling the i-th index of the first mode, and we simply sample without replacement from the set $\{1 \cdots I\}$, using $p_{\mathcal{I}}$ as bias. The very same idea is used for matrices $\mathbf{Y}_i$. Doing so is preferable over sampling uniformly, since our bias makes it more probable that high density indices of the data will be retained on the sample, and hence, it will be more representative of the entire set.

Suppose that we call $\mathcal{I}, \mathcal{J}, \mathcal{K}$ the index samples for the three modes of $\underline{\mathbf{X}}$. Then, we may take $\underline{\mathbf{X}}_s = \underline{\mathbf{X}}(\mathcal{I}, \mathcal{J}, \mathcal{K})$ (and similarly for matrices $\mathbf{Y}_i$); essentially, what we are left with is a small, yet representative, sample of our original dataset, where the high density blocks are more likely to appear on the sample. It is important to note that the indices of the coupled modes are the same for the matrix and the tensor, e.g. $\mathbf{I}$ randomly selects the same set of indices for $\underline{\mathbf{X}}$ and $\mathbf{Y}_1$. This way, we make sure that the coupling is *preserved* after sampling.

Phase 2: Fit CMTF to reduced data Having said that, the key idea of our proposed algorithm is to run ALS CMTF (Algorithm 1) on the sample and then, based on the sampled indices, redistribute the result to the original index space. In more detail, suppose that $\mathbf{A}_s$ is the factor matrix obtained by the aforementioned procedure, and that jointly describes the first mode of $\underline{\mathbf{X}}_s$ and $\mathbf{Y}_{1,s}$. The dimensions of $\mathbf{A}_s$ are going to be $|\mathcal{I}| \times F$ (where $|\dot{|}$ denotes cardinality and F is the number of factors). Let us further assume matrix $\mathbf{A}$ of size $I \times F$ which expresses the first mode of the tensor and the matrix, before sampling; due to sampling, it holds that $I \gg |\mathcal{I}|$. If we initially set all entries of $\mathbf{A}$ to zero and we further set $\mathbf{A}(\mathcal{I}, :) = \mathbf{A}_s$ we obtain a highly *sparse* factor matrix whose non-zero values are a 'best effort' approximation of the true ones, i.e. the values of the factor matrix that we would obtain by decomposing the full data.

So far, we have provided a description of the algorithm where only one repetition of sampling is used. However, if our sample consists of only a small portion of the data, inevitably, this will not be adequate in order to successfully model all variation in the data. To that end, we allow for multiple sampling repetitions in our algorithm, i.e. extracting multiple sample tensors $\underline{\mathbf{X}}_s$ and side matrices $\mathbf{Y}_{i,s}$, fitting a CMTF model to all of them and combining the results in a way that the true latent patterns are retained. We are going to provide a detailed outline of how to carry the multi-repetition version of SCOUP-SMT in the following.

While doing multiple repetitions, we keep a *common.* subset of indices for all different samples. In particular, let p be the percentage of common values across all repetitions and $\mathcal{I}_p$ denote the common indices along the first mode (same notation applies to the rest of the indices); then, all sample tensors $\underline{\mathbf{X}}_s$ will definitely contain the indices $\mathcal{I}_p$ on the first mode, as well as $(1-p)\frac{I}{s}$ indices sampled independently (across repetitions) at random. This common index sample is key in order to ensure that our results are not rank deficient, and all partial results are merged correctly.

We do not provide an exact method for choosing p, however, as a rule of thumb, we observed that, depending on how sparse and noisy the data is, a range of p between 0.2 and 0.5 works well. This introduces a trade-off between redundancy of indices that we sample, versus the accuracy of the decomposition; since we are not dealing solely with tensors, which are known to be well behaved in terms of decomposition uniqueness, it pays off to introduce some data redundancy (especially when SCOUP-SMT runs in a parallel system) so that we avoid rank-deficiency in our data.

Let r be the number of different sampling repetitions, resulting

$$\mathbf{x}_A(i) = \sum_{j=1}^{J}\sum_{k=1}^{K} |\underline{\mathbf{X}}(i,j,k)| + \sum_{j=1}^{I_1} |\mathbf{Y}_1(i,j)|,\ \mathbf{x}_B(j) = \sum_{i=1}^{I}\sum_{k=1}^{K} |\underline{\mathbf{X}}(i,j,k)| + \sum_{i=1}^{I_2} |\mathbf{Y}_2(j,i)|,\ \mathbf{x}_C(k) = \sum_{i=1}^{I}\sum_{j=1}^{J} |\underline{\mathbf{X}}(i,j,k)| + \sum_{j=1}^{I_3} |\mathbf{Y}_3(k,j)|, \tag{2}$$

$$\mathbf{y}_{1,A}(i) = \sum_{j=1}^{I_1} |\mathbf{Y}_1(i,j)|\ \mathbf{y}_{2,B}(j) = \sum_{i=1}^{I_2} |\mathbf{Y}_2(j,i)|,\ \mathbf{y}_{3,C}(k) = \sum_{j=1}^{I_3} |\mathbf{Y}_3(k,j)| \tag{3}$$

$$\mathbf{y}_{1,D}(j) = \sum_{i=1}^{I} |\mathbf{Y}_1(i,j)|,\ \mathbf{y}_{2,G}(i) = \sum_{j=1}^{J} |\mathbf{Y}_2(j,i)|,\ \mathbf{y}_{3,E}(i) = \sum_{k=1}^{K} |\mathbf{Y}_3(k,i)| \tag{4}$$

in r different sets of sampled matrix-tensor couples $\underline{\mathbf{X}}_s^{(i)}$ and $\mathbf{Y}_{j,s}^{(i)}$ ($i = 1\cdots r,\ j = 1\cdots 3$). For that set of coupled data, we fit a CMTF model, using Algorithm 1, obtaining a set of factor matrices $\mathbf{A}^{(i)}$ (and likewise for the rest).

Phase 3: Merging partial results After having obtained these r different sets of partial results, as a final step, we have to merge them together into a set of factor matrices that we would ideally get had we operated on the full dataset.

In order to make the merging work, we first introduce the following scaling on each column of each factor matrix: Let's take $\mathbf{A}^{(i)}$ for example; we normalize each column of $\mathbf{A}$ by the ℓ_2 norm of the common part, as described in line 8 of Algorithm 2. By doing so, the common part of each factor matrix (for all repetitions) will be unit norm. This scaling is absorbed in a set of scaling vectors $\boldsymbol{\lambda}_A$ (and accordingly for the rest of the factors). The new objective function is shown in Equation 5

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D},\mathbf{E},\mathbf{G}} \|\underline{\mathbf{X}} - \sum_k \boldsymbol{\lambda}_A(k)\boldsymbol{\lambda}_B(k)\boldsymbol{\lambda}_C(k)\mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k\|_F^2 + \tag{5}$$
$$\|\mathbf{Y}_1 - \mathbf{A}\ \mathrm{diag}(\boldsymbol{\lambda}_A * \boldsymbol{\lambda}_D)\ \mathbf{D}^T\|_F^2 + \|\mathbf{Y}_2 - \mathbf{B}\ \mathrm{diag}(\boldsymbol{\lambda}_B * \boldsymbol{\lambda}_E)\ \mathbf{E}^T\|_F^2$$
$$+ \|\mathbf{Y}_3 - \mathbf{C}\ \mathrm{diag}(\boldsymbol{\lambda}_C * \boldsymbol{\lambda}_G)\ \mathbf{G}^T\|_F^2$$

A problem that is introduced by carrying out multiple sampling repetitions is that the correspondence of the output factors of each repetition is very likely to be distorted. In other words, say we have matrices $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ and we wish to merge their columns (i.e. the latent components) into a single matrix $\mathbf{A}$, by stitching together columns that correspond to the same component. It might very well be the case that the order in which the latent components appear in $\mathbf{A}^{(1)}$ is not the same as in $\mathbf{A}^{(2)}$.

The sole purpose of the aforementioned normalization is to resolve the correspondence problem. In Algorithm 3, we merge the partial results while establishing the correct correspondence of the columns. Theoretical intuition as to why this is possible follows as a proof sketch:

Following the example of $r = 2$ of the previous paragraph, according to Algorithm 3, we compute the inner product of the common parts of each column of $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$. Since the common parts of each column are normalized to unit norm, then the inner product of the common part of the column of $\mathbf{A}^{(1)}$ with that of $\mathbf{A}^{(2)}$ will be maximized (and exactly equal to 1) for the matching columns, and by the Cauchy-Schwartz inequality, for all other combinations, it will be less than 1.

3.2 Speeding up the core of the algorithm

In addition to our main contribution in terms of speeding up the decomposition, i.e. Algorithm 2, we are able to further speed the algorithm up, by making a few careful interventions to the core algorithm (Algorithm 1).

LEMMA 1. *We may do the following simplification to each pseudoinversion step of the ALS algorithm (Algorithm 1):*

$$\begin{bmatrix}\mathbf{A}\odot\mathbf{B}\\ \mathbf{M}\end{bmatrix}^{\dagger} = \left(\mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B} + \mathbf{M}^T * \mathbf{M}\right)^{\dagger}\left[(\mathbf{A}\odot\mathbf{B})^T, \mathbf{M}^T\right]$$

PROOF. For the Moore-Penrose pseudoinverse of the Khatri-Rao product, it holds that [7], [14]

$$(\mathbf{A}\odot\mathbf{B})^{\dagger} = \left(\mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B}\right)^{\dagger}(\mathbf{A}\odot\mathbf{B})^T$$

Furthermore [7]

$$(\mathbf{A}\odot\mathbf{B})^T(\mathbf{A}\odot\mathbf{B}) = \mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B}$$

For a partitioned matrix $\mathbf{P} = \begin{bmatrix}\mathbf{P}_1\\ \mathbf{P}_2\end{bmatrix}$, it holds that its pseudoinverse may be written in the following form [9]

$$\begin{bmatrix}\mathbf{P}_1\\ \mathbf{P}_2\end{bmatrix}^{\dagger} = \left(\mathbf{P}_1^T\mathbf{P}_1 + \mathbf{P}_2^T\mathbf{P}_2\right)^{\dagger}\left[\mathbf{P}_1^T,\quad \mathbf{P}_2^T\right]$$

Putting things together, it follows:

$$\begin{bmatrix}\mathbf{A}\odot\mathbf{B}\\ \mathbf{M}\end{bmatrix}^{\dagger} = \left(\mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B} + \mathbf{M}^T * \mathbf{M}\right)^{\dagger}\left[(\mathbf{A}\odot\mathbf{B})^T, \mathbf{M}^T\right]$$

which concludes the proof. □

The above lemma implies that substituting the naive pseudoinversion of $\begin{bmatrix}\mathbf{A}\odot\mathbf{B}\\ \mathbf{M}\end{bmatrix}$ with the simplified version, offers significant *computational* gains to Algorithm 1 and hence to Algorithm 2. More precisely, if the dimensions of $\mathbf{A}, \mathbf{B}$ and $\mathbf{M}$ are $I \times R$, $J \times R$ and $I \times I_2$, then computing the pseudoinverse naively would cost $O\left(R^2\left(IJ + I_2\right)\right)$, whereas our proposed method yields a cost of $O\left(R^2\left(I + J + I_2\right)\right)$ because of the fact that we are pseudoinverting only a *small* $R \times R$ matrix. We have to note here that in almost all practical scenarios $R \ll I, J, I_2$.

3.3 Accounting for missing values

In many practical scenarios, we often have corrupted or missing data. For instance, when measuring brain activity, a few sensors might stop working, whereas the majority of the sensors produce useful signal. Despite these common data imperfections, it is important for a data mining algorithm to be able to operate.

We carefully ignore the missing values from the entire optimization procedure: Notice that is *not* the same as simply zeroing out all missing values, since 0 might have a valid physical interpretation. Specifically, we define a 'weight' tensor $\underline{\mathbf{W}}$ which has '0' in all coefficients where values are missing, and '1' everywhere else. Similarly, we introduce three weight matrices $\mathbf{W}_i$ for each of the coupled matrices $\mathbf{Y}_i$. Then, the optimization function of the CMTF model becomes

Algorithm 2: SCOUP-SMT: Fast, sparse, and parallel CMTF

Input: Tensor $\underline{\mathbf{X}}$ of size $I \times J \times K$, matrices $\mathbf{Y}_i,\ i = 1 \cdots 3$, of size $I \times I_2$, $J \times J_2$, and $K \times K_2$ respectively, number of factors F, sampling factor s, number of repetitions r.

Output: $\mathbf{A}$ of size $I \times F$, $\mathbf{b}$ of size $J \times F$, $\mathbf{c}$ of size $K \times F$, $\mathbf{D}$ of size $I_2 \times F$, $\mathbf{G}$ of size $J_2 \times F$, $\mathbf{E}$ of size $K_2 \times F$. $\boldsymbol{\lambda}_A$, $\boldsymbol{\lambda}_B$, $\boldsymbol{\lambda}_C$, $\boldsymbol{\lambda}_D$, $\boldsymbol{\lambda}_E$, $\boldsymbol{\lambda}_G$ of size $F \times 1$ which contains the scale of each component for each factor matrix.

1: Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{G}$ to all-zeros.
2: Randomly, *using mode densities as bias*, select a set of $100p\%$ ($p \in [0, 1]$) indices $\mathcal{I}_p, \mathcal{J}_p, \mathcal{K}_p$ to be common across all repetitions. For example, $\mathcal{I}_p$ is sampled with probabilities with $p_{\mathcal{I}}(i) = \mathbf{x}_A(i) / \sum_{i=1}^{I} \mathbf{x}_A(i)$. Probabilities for the rest of the modes are calculated similarly.
3: **for** $i = 1 \cdots r$ **do**
{**Phase 1: Sample indices**}
4: Compute densities as in equations 2, 3, 4. Compute set of indices $\mathcal{I}^{(i)}$ as random sample without replacement of $\{1 \cdots I\}$ of size $I/(s(1-p))$ with probability $p_{\mathcal{I}}(i) = \mathbf{x}_A(i) / \sum_{i=1}^{I} \mathbf{x}_a(i)$. Likewise for $\mathcal{J}, \mathcal{K},$, $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{I}_3$. Set $\mathcal{I}^{(i)} = \mathcal{I} \cup \mathcal{I}_p$. Likewise for the rest.
5: Get $\underline{\mathbf{X}}_s^{(i)} = \underline{\mathbf{X}}(\mathcal{I}^{(i)}, \mathcal{J}^{(i)}, \mathcal{K}^{(i)})$, $\mathbf{Y}_{1s}^{(i)} = \mathbf{Y}_1(\mathcal{I}^{(i)}, \mathcal{I}_1^{(i)})$ and likewise for $\mathbf{Y}_{2s}^{(i)}$ and $\mathbf{Y}_{3s}^{(i)}$. Note that the same index sample is used for *coupled* modes.
{**Phase: Fit the model on the sampled data**}
6: Run Algorithm 1 for $\underline{\mathbf{X}}_s^{(i)}$ and $\mathbf{Y}_{js}^{(i)},\ j = 1 \cdots 3$ and obtain $\mathbf{A}_s, \mathbf{B}_s, \mathbf{C}_s, \mathbf{D}_s, \mathbf{G}_s, \mathbf{E}_s$.
7: $\mathbf{A}^{(i)}(\mathcal{I}^{(i)}, :) = \mathbf{A}_s$. Likewise for the rest.
8: Calculate the ℓ_2 norm of the columns of the common part: $\boldsymbol{\lambda}_A^{(i)}(f) = \|\mathbf{A}^{(i)}(\mathcal{I}_p, f)\|_2$, for $f = 1 \cdots F$. Normalize columns of $\mathbf{A}^{(i)}$ using $\boldsymbol{\lambda}_A^{(i)}$ (likewise for the rest). Note that the common part of each factor will now be normalized to unit norm.
9: **end for**
{**Phase 3: Merge partial results**}
10: $\mathbf{A}$ =MERGE $(\mathbf{A}_1^i)$. Likewise for the rest.
11: $\boldsymbol{\lambda}_A$ = average of $\boldsymbol{\lambda}_{A1}^i$. Likewise for the rest.

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D},\mathbf{E},\mathbf{G}} \|\underline{\mathbf{W}} * \left(\underline{\mathbf{X}} - \sum_k \mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k\right)\|_F^2 + \|\mathbf{W}_1 * \left(\mathbf{Y}_1 - \mathbf{A}\mathbf{D}^T\right)\|_F^2 + \|\mathbf{W}_2 * \left(\mathbf{Y}_2 - \mathbf{B}\mathbf{E}^T\right)\|_F^2 + \|\mathbf{W}_3 * \left(\mathbf{Y}_3 - \mathbf{C}\mathbf{G}^T\right)\|_F^2$$

As we show in Algorithm 1, we may solve CMTF by solving six least squares problems in an alternating fashion. A fortuitous implication of this fact is that in order to handle missing values for CMTF, it suffices to solve

$$\min_{\mathbf{B}} \|\mathbf{W} * \left(\mathbf{X} - \mathbf{A}\mathbf{B}^T\right)\|_F^2 \tag{6}$$

where $\mathbf{W}$ is a weight matrix in the same sense as described a few lines earlier.

On our way tackling the above problem, we first need to investigate its scalar case, i.e. the case where we are interested only in $\mathbf{B}(j, f)$ for a fixed pair of j and f. The optimization problem may

Algorithm 3: MERGE: Given partial results of factor matrices, merge them correctly

Input: Factor matrices $\mathbf{A}_1^i$ of size $I \times F$ each, and r is the number of repetitions, $\mathcal{I}_p$: set of common indices.

Output: Factor matrix $\mathbf{A}$ of size $I \times F$.

1: Set $\mathbf{A} = \mathbf{A}^{(1)}$
2: Set $\ell = \{1 \cdots F\}$, a list that keeps track of which columns have not been assigned yet.
3: **for** $i = 2 \cdots r$ **do**
4: **for** $f_1 = 1 \cdots F$ **do**
5: **for** f_2 in ℓ **do**
6: Compute similarity $\mathbf{v}(f_2) = (\mathbf{A}(\mathcal{I}_p, f_2))^T \left(\mathbf{A}^{(i)}(\mathcal{I}_p, f_1))\right)$
7: **end for**
8: $c^* = \arg\max_c \mathbf{v}(c)$ (Ideally, for the matching columns, the inner product should be close to 1; conversely, for the rest of the columns, it should be considerably smaller)
9: $\mathbf{A}(:, c^*) = \mathbf{A}^{(i)}(:, f_1)\Big|_{\mathbf{A}(:,c^*)=0}$, i.e. update the zero entries of the column.
10: Remove c^* from list ℓ.
11: **end for**
12: **end for**

be rewritten as

$$\min_{\mathbf{B}(j,f)} \|\mathbf{W}(:, j) * \mathbf{X}(:, j) - (\mathbf{W}(:, j) * \mathbf{A}(: f))\, \mathbf{B}(j, f)^T\|$$

which is essentially a scalar least squares problem of the form:

$$\min_b \|\mathbf{x} - \mathbf{a}b\|_2^2$$

with solution in analytical form: $b = \frac{\mathbf{x}^T \mathbf{a}}{\|\mathbf{a}\|_2^2}$

We may, thus, solve this problem of Equation 6 using *element-wise coordinate descent*, where we update each coefficient of $\mathbf{B}$ iteratively, until convergence. Therefore, with the aforementioned derivation, we are able to modify our original algorithm in order to take missing values into account.

3.4 Parallelization

Our proposed algorithm is, by its nature, parallelizable; in essence, we generate multiple samples of the coupled data, we fit a CMTF model to each sample and then we merge the results. By carefully observing Algorithm 2, we can see that lines 3 to 9 may be carried out entirely in parallel, provided that we have a good enough random number generator that does not generate the very same sample across all r repetitions. In particular, the r repetitions are independent from one another, since computing the set of common indices (line 2), which is the common factor across all repetitions, is done before line 3.

4. KNOWLEDGE DISCOVERY

4.1 Scoup-SMT on Brain Image Data With Additional Semantic Information

As part of a larger study of neural representations of word meanings in the human brain [16], we applied Scoup-SMT to a combination of datasets which we henceforth jointly refer to as BRAINQ. This dataset consists of two parts. The first is a tensor that contains measurements of the fMRI brain activity of 9 human subjects, when shown each of 60 concrete nouns (5 in each of 12 categories, e.g. dog, hand, house, door, shirt, dresser, butterfly, knife, telephone, saw, lettuce, train). fMRI measures slow changes in blood oxygenation levels, reflecting localized changes in brain activity.

Here our data is made up of $3 \times 3 \times 6$mm voxels (3D pixels) corresponding to fixed spatial locations across participants. Recorded fMRI values are the mean activity over 4 contiguous seconds, averaged over multiple presentations of each stimulus word (each word is presented 6 times as a stimulus). Further acquisition and preprocessing details are given in [16]. This dataset is publicly available[2]. The second part of the data is a matrix containing answers to 218 questions pertaining to the semantics of these 60 nouns. A sample of these questions is shown in Table 2.

This dataset has been used before in works such as [17], [18].

BRAINQ's size is $60 \times 77775 \times 9$ with over 11 million non-zeros (tensor), and 60×218 with about 11.000 non-zeros (matrix). The dimensions might not be extremely high, however, the data is *very dense* and it is therefore difficult to handle efficiently. For instance, decomposing the dataset using the simple ALS algorithm took more than 24 hours, whereas SCOUP-SMT yielded a speedup of 50-100$\times$ over this (cf. Figure 1).

Simultaneous Clustering of Words, Questions and Regions of the Brain

One of the strengths of our proposed method is its expressiveness in terms of simultaneously soft-clustering all involved entities of the problem. By taking a low rank decomposition of the BRAINQ data (using $r = 5$ and $s_I = 3,\ s_J = 86,\ s_K = 1$ for the tensor and s_I for the questions dimension of the matrix)[3], we are able to find groups that jointly express words, questions and brain voxels (we can also derive groups of human subjects; however, it is an active research subject in neuroscience, whether brain-scans should differ significantly between people, and is out of the scope of the present work).

In Figure 4, we display 4 such groups of brain regions that are activated given a stimulus of a group of similar words; these words can be seen in Table 2, along with groups of similar questions that were highly correlated with the words of each group. Moreover, we were able to successfully identify high activation of the *premotor cortex* in Group 3, which is associated with concepts such as holding or picking items up.

	Nouns	**Questions**
Group1	beetle	can it cause you pain?
	pants	do you see it daily?
	bee	is it conscious?
Group 2	bear	does it grow?
	cow	is it alive?
	coat	was it ever alive?
Group 3	glass	can you pick it up?
	tomato	can you hold it in one hand?
	bell	is it smaller than a golfball?'
Group 4	bed	does it use electricity?
	house	can you sit on it?
	car	does it cast a shadow?

Table 2: Groups of nouns and questions that are both positively and negatively correlated, and correspond to the brain regions show in Fig. 4.

By-product: Predicting Brain Activity from Questions

In addition to soft-clustering, the low rank joint decomposition of the BRAINQ data offers another significant result. This low dimensional embedding of the data into a common semantic space, enables the prediction of, say, the brain activity of a subject, for a given word, given the corresponding vector of question answers for that word. In particular, by projecting the question answer vector to the latent semantic space and then expanding it to the brain voxel space, we obtain a fairly good prediction of the brain activity.

To evaluate the accuracy of these predictions of brain activity, we follow a *leave-two-out* scheme, where we remove two words entirely from the brain tensor and the question matrix; we carry out the joint decomposition, in some very low dimension, for the remaining set of words and we obtain the usual set of matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$. Due to the randomized nature of SCOUP-SMT, we did 100 repetitions of the procedure described below.

Let $\mathbf{q}_i$ be the question vector for some word i, and $\mathbf{v}_i$ be the brain activity of one human subject, pertaining to the same word. By left-multiplying $\mathbf{q}_i$ with $\mathbf{D}^T$, we project $\mathbf{q}_i$ to the latent space of the decomposition; then, by left-multiplying the result with $\mathbf{B}$, we project the result to the brain voxel space. Thus, our estimated (predicted) brain activity is obtained as $\hat{\mathbf{v}}_i = \mathbf{B}\mathbf{D}^T\mathbf{q}_i$

Given the predicted brain activities $\hat{\mathbf{v}}_\mathbf{1}$ and $\hat{\mathbf{v}}_\mathbf{2}$ for the two left out words, and the two actual brain images $\mathbf{v_1}$ and $\mathbf{v_2}$ which were withheld from the training data, the *leave-two-out* scheme measures prediction accuracy by the ability to choose which of the observed brain images corresponds to which of the two words. After mean-centering the vectors, this classification decision is made according to the following rule:

$$\|\mathbf{v}_1 - \hat{\mathbf{v}}_1\|_2 + \|\mathbf{v}_2 - \hat{\mathbf{v}}_2\|_2 < \|\mathbf{v}_1 - \hat{\mathbf{v}}_2\|_2 + \|\mathbf{v}_2 - \hat{\mathbf{v}}_1\|_2$$

Although our approach is not designed to make predictions, preliminary results are very encouraging: Using only F=2 components, for the noun pair *closet/watch* we obtained mean accuracy of about 0.82 for 5 out of the 9 human subjects. Similarly, for the pair *knife/beetle*, we achieved accuracy of about 0.8 for a somewhat different group of 5 subjects. For the rest of the human subjects, the accuracy is considerably lower, however, it may be the case that brain activity predictability varies between subjects, a fact that requires further investigation.

We plan detailed experiments to determine the accuracy of these predictions compared to specialized methods that have previously been used for these predictions, but which do not have the ability of our method to discover latent representations, such as [18].

4.2 Generality: Mining Social Networks with Additional Information

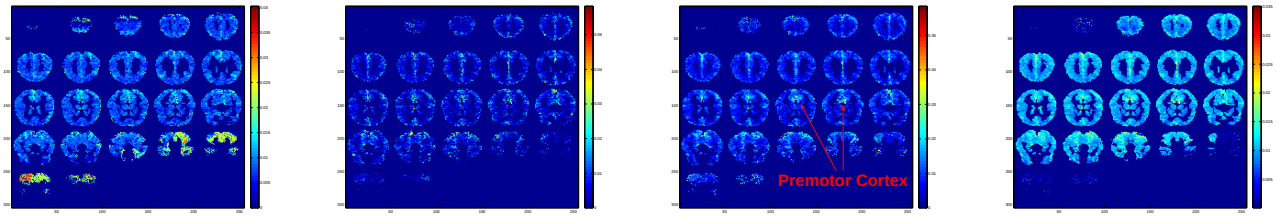
We have demonstrated the expressive power of SCOUP-SMT for the BRAINQ dataset, but in this subsection, we stress the fact that the method is actually application independent and may be used in vastly different scenarios. To that end, we analyze a FACEBOOK dataset, introduced in [22][4]. This dataset consists of a $63890 \times 63890 \times 1847$ (wall, poster, day) tensor with about 740.000 non-zeros, and a 63890×63890 who is friends with whom matrix, with about 1.6 million non-zeros. In contrast to BRAINQ, this dataset is very sparse (as one would expect from a social network dataset). However, SCOUP-SMT works in both cases, demonstrating that it can analyze data efficiently, regardless of their density.

We decomposed the data into 25 rank one components, using $s_I = 1000,\ s_J = 1000, s_K = 100$ and s_I for both dimensions of the matrix, and manually inspected the results. A fair amount of components captured normal activity of Facebook users who occasionally post on their friends' walls; here we only show one outstanding anomaly, due to lack of space: In Fig. 5 we show what appears to be a spammer, i.e. a person who, only on a certain day, posts on many different friends' walls: the first subfigure corresponds to the wall owners, the second subfigure corresponds to the

[2] `http://www.cs.cmu.edu/afs/cs/project/theo-73/www/science2008/data.html`

[3] We may use imbalanced sampling factors, especially when the data is far from being 'rectangular'.

[4] Download FACEBOOK at `http://socialnetworks.mpi-sws.org/dat`



(a) Group 1 (b) Group 2 (c) Group 3 (d) Group 4

Figure 4: The latent brain images for the 4 word/question groups as shown in Table 2. We can see that for each different group, the activation pattern of certain brain regions is different. For instance, Group 3 refers to small items that can be held in one hand,such as a tomato or a glass, and the activation pattern is very different from the one of Group 1, which mostly refers to insects, such as bee or beetle. Additionally, Group 3, for instance, shows high activation in the *premotor cortex* which is associated with the concepts of that group.

people who post on these walls, and the third subfigure is the time (measured in days); we thus have one person, posting on many peoples' walls, on a single day.

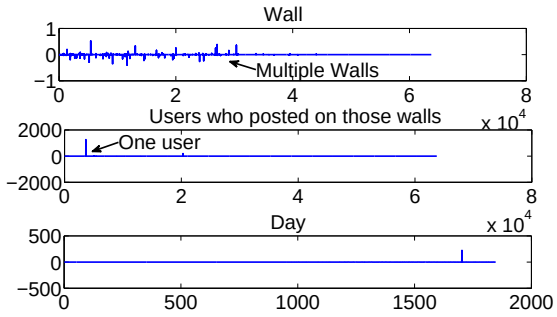


Figure 5: This is a pattern extracted using SCOUP-SMT, which shows what appears to be a spammer on the FACEBOOK dataset: One person, posting to many different walls on a single day.

5. EXPERIMENTS

We implemented SCOUP-SMT in Matlab. For the parallelization of the algorithm, we used Matlab's Parallel Computing Toolbox. For tensor manipulation, we used the Tensor Toolbox for Matlab [6] which is optimized especially for sparse tensors (but works very well for dense ones too). All experiments were carried out on a machine with 2 dual-core AMD Opteron 880 processors (2.4 GHz), 4 TB disk, and 48GB ram. The parallel experiments were run on all 4 cores, which justifies our choice of $r = 4$ in this case. Whenever we conducted multiple iterations of an experiment (due to the randomized nature of SCOUP-SMT), we report error-bars along the plots. For all the following experiments we used either portions of the BRAINQ dataset, or the whole dataset.

5.1 Accuracy

In Figure 6 we demonstrate that the algorithm operates correctly, in the sense that it reduces the model cost (Equation 1) when doing more repetitions. In particular, the vertical axis displays the relative cost, i.e. $\frac{\text{SCOUP-SMT cost}}{\text{ALS cost}}$ (with ideal being equal to 1) and the horizontal axis is the number of repetitions in the sampling. We observed that for a few executions of the algorithm, the cost was not monotonically decreasing; however, we ran the algorithm 1000 times, keeping the executions that decreased the relative cost monotonically and plotted them in Fig. 6.

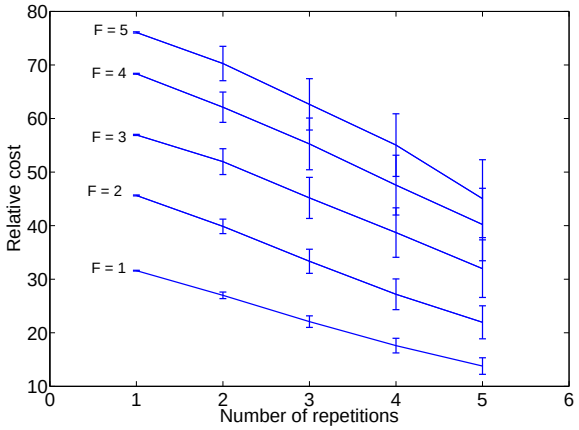


Figure 6: The relative cost of the model, as a function of the number of repetitions r is decreasing, which empirically shows that SCOUP-SMT actually reduces the approximation error of the CMTF model.

5.2 Speedup

As we have already discussed in the Introduction, SCOUP-SMT achieves a speedup of 50-100 on the BRAINQ; for the $50\times$ case, the same approximation error of the CMTF objective is maintained, while for higher speedup values, the relative cost increases, but within reasonable range. Figure 1 illustrates this behaviour.

Additionally, SCOUP-SMT benefits greatly from it's inherent parallelizability. The parallel results we report come from $r = 4$ repetitions of sampling, carried out on 4 cores; had more cores been available, we would probably observe a higher speedup (keeping of course Amdahl's law in mind), while maintaining low relative cost, since we establish in the previous subsection that the more repetitions we do, the better we approximate the CMTF model.

5.3 Sparsity

One of the main advantages of SCOUP-SMT is that, *by construction*, it produces *sparse* latent factors for coupled matrix-tensor model. In Fig. 7 we demonstrate the sparsity of SCOUP-SMT's results by introducing the relative sparsity metric; this intuitive metric

is simply the ratio of the output size of the ALS algorithm, divided by the output size of SCOUP-SMT. The output size is simply calculated by adding up the number of non-zero entries for all factor matrices output by the algorithm. We use a portion of the BRAINQ dataset in order to execute this experiment. We can see that for the dense BRAINQ dataset, we obtained twice as sparse results. However, in experiments with randomly generated, sparse, data, we experienced higher degrees of sparsity, in the order of $5\times$. We omit such plots due to space constraints.

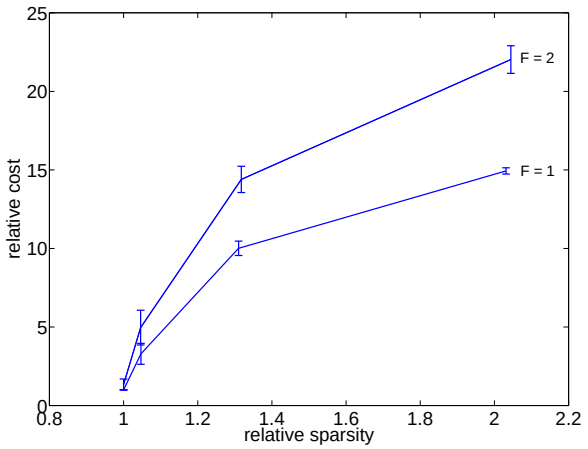


Figure 7: The relative output size vs. the relative cost indicates that, even for very dense datasets such as BRAINQ, we are able to get a 2 fold decrease in the output size, while maintaining good approximation cost.

5.4 Robustness to missing values

In order to measure resilience to missing values we define the *Signal-to-Noise Ratio* (SNR) as simply as SNR $= \frac{\|\underline{\mathbf{X}}_m\|_F^2}{\|\underline{\mathbf{X}}_m - \underline{\mathbf{X}}_0\|_F^2}$, where $\underline{\mathbf{X}}_m$ is the reconstructed tensor when a m fraction of the values are missing. In Figure 8, we demonstrate the results of that experiment; we observe that even for a fair amount of missing data, the algorithm performs reasonably well, achieving high SNR. Moreover, for small amounts of missing data, the speed of the algorithm is not degraded, while for larger values, it is considerably slower, probably due to Matlab's implementation issues. However, this is encouraging, in the sense that if the amount of missing data is not overwhelming, SCOUP-SMT is able to deliver a very good approximation of the latent subspace. This experiment was, again, conducted on a portion of BRAINQ.

6. RELATED WORK

Coupled, Multi-block, Multi-set Models Coupled Matrix-Tensor Factorizations belong to a family of models also referred to as *Multi-block* or *Multi-set* in the literature. Smilde et al. in [20] provided the first disciplined treatment of such multi-block models, in a chemometric context. An important issue with these models is how to weigh the different data blocks such that scaling differences may be alleviated. In [23], Wilderjans et al. propose and compare two different weighing schemes. Most related to the present work is the work of Acar et al. in [4], where a first order optimization approach is proposed, in order to solve the CMTF problem. As we mention in the Introduction, SCOUP-SMT is compatible with this algorithm, since it provides an alternative to the core CMTF solver.

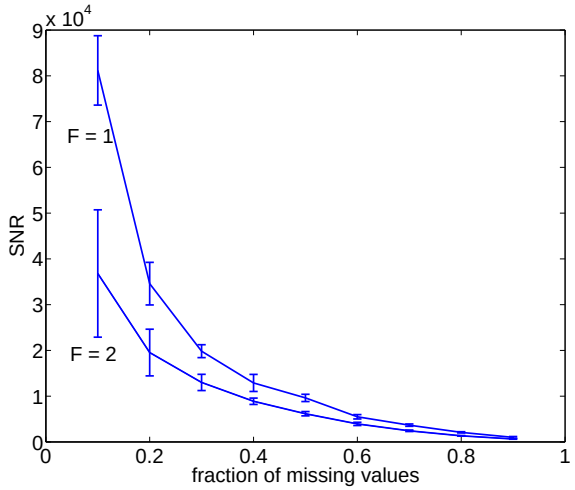


Figure 8: This Figure shows the Signal-to-Noise ratio (SNR)-as defined in the main text- as a function of the percentage of missing values. We can observe that, even for a fair amount of missing values, the SNR is quite high, signifying that SCOUP-SMT is able to handle such ill-conditioned settings, with reasonable fidelity.

In [5], Acar et. al apply the CMTF model, using the aforementioned first-order approach in a chemometrics setting. In [3], Acar et. al introduce a coupled matrix decomposition, where two matrices match on one of the two dimensions, and are decomposed in the same spirit as in CMTF, while imposing explicit sparsity constraints (via ℓ_1 norm penalties); although SCOUP-SMT also produces sparse factors, this so happens as a fortuitous byproduct of sampling, whereas in [3] an explicit sparsity penalty is considered. As an interesting application, in [26], the authors employ CMTF for Collaborative Filtering. On a related note, [24], [13], and [15] introduce models where multiple tensors are coupled with respect to one mode, and analyzed jointly; in this work, we don't consider coupling of two (or more) tensors, however, we leave that for future work.

Having listed an outline of relevant approaches, to the best of our knowledge, SCOUP-SMT is the first algorithm for CMTF that combines speed, parallelization, as well as sparse factors. An alternative perspective on SCOUP-SMT is that of a framework that is able to speed up and sparsify any (possibly highly fine tuned) core algorithm for CMTF.

Treating Missing Values in Tensor Decompositions Tomasi et. al [21] provides a very comprehensive study on how to handle missing values for plain tensor decompositions.

Fast & Scalable Tensor Decompositions In [19] we introduced a parallel algorithm for the regular PARAFAC decomposition, where a sampling scheme of similar nature as here is exploited; in [10], a scalable MapReduce implementation of PARAFAC is presented. Additionally, the mechanics behind the Tensor Toolbox for Matlab [6] are very powerful when it comes to memory-resident tensors. Finally, in [25], the authors introduce a parallel framework in order to handle tensor decompositions efficiently.

Tensor applications to brain data There has been substantial related work, which utilizes tensors for this purpose, e.g. [8], [2].

7. CONCLUSIONS

Our main contributions are the following:

- *Fast, parallel & sparsity promoting algorithm:* SCOUP-SMT

is up to *50-100 times* faster than state of the art algorithms.
- *Robustness to missing data*: SCOUP-SMT can effectively handle missing values, without significant performance degradation, even for moderate amounts of missing entries.
- *Effectiveness and Knowledge Discovery*: SCOUP-SMT, applied to the BRAINQ dataset, discovers meaningful triple-mode clusters: clusters of words, of questions, and of brain regions have similar behavior; as a by-product, SCOUP-SMT is able to predict brain activity with very promising preliminary results.
- *Generality*: We applied SCOUP-SMT to a FACEBOOK dataset with additional information, identifying what appears to be a spammer.

Acknowledgements

Research was funded by grants NSF IIS-1247489, NSF IIS-1247632, NSF CDI 0835797, and NIH/NICHD 12165321. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties. The authors would also like to thank Leila Wehbe and Alona Fyshe for their initial help with the BRAINQ data.

8. REFERENCES

[1] Read the web. `http://rtw.ml.cmu.edu/rtw/`.

[2] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener. Multiway analysis of epilepsy tensors. *Bioinformatics*, 23(13):i10–i18, 2007.

[3] E. Acar, G. Gurdeniz, M.A. Rasmussen, D. Rago, L.O. Dragsted, and R. Bro. Coupled matrix factorization with sparse factors to identify potential biomarkers in metabolomics. In *IEEE ICDM Workshops*, pages 1–8. IEEE, 2012.

[4] E. Acar, T.G. Kolda, and D.M. Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv preprint arXiv:1105.3422*, 2011.

[5] E. Acar, G.E. Plopper, and B. Yener. Coupled analysis of in vitro and histology tissue samples to quantify structure-function relationship. *PloS one*, 7(3):e32227, 2012.

[6] B.W. Bader and T.G. Kolda. Matlab tensor toolbox version 2.2. *Albuquerque, NM, USA: Sandia National Laboratories*, 2007.

[7] R. Bro. *Multi-way analysis in the food industry: models, algorithms, and applications*. PhD thesis, Københavns Universitet, 1998.

[8] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. Wiley, 2009.

[9] C. Hung and T.L. Markham. The moore-penrose inverse of a partitioned matrix m= adbc. *Linear Algebra and its Applications*, 11(1):73–86, 1975.

[10] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In *SIGKDD*, pages 316–324. ACM, 2012.

[11] H.A.L. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14(3):105–122, 2000.

[12] T.G. Kolda and B.W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3), 2009.

[13] Y.R. Lin, J. Sun, P. Castro, R. Konuru, H. Sundaram, and A. Kelliher. Metafac: community discovery via relational hypergraph factorization. In *SIGKDD*, pages 527–536. ACM, 2009.

[14] S. Liu and G. Trenkler. Hadamard, khatri-rao, kronecker and other matrix products. *International Journal of Information and Systems Sciences*, pages 160–177, 2008.

[15] W. Liu, J. Chan, J. Bailey, C. Leckie, and K. Ramamohanarao. Mining labelled tensors by discovering both their common and discriminative subspaces. In *SDM 2013*. SIAM, 2013.

[16] T.M. Mitchell, S.V. Shinkareva, A. Carlson, K.M. Chang, V.L. Malave, R.A. Mason, and M.A. Just. Predicting human brain activity associated with the meanings of nouns. *Science*, 320(5880):1191–1195, 2008.

[17] Brian Murphy, Partha Talukdar, and Tom Mitchell. Selecting corpus-semantic models for neurolinguistic decoding. In *First Joint Conference on Lexical and Computational Semantics (* SEM)*, pages 114–123, 2012.

[18] Mark Palatucci, Dean Pomerleau, Geoffrey Hinton, and Tom Mitchell. Zero-shot learning with semantic output codes. *Advances in neural information processing systems*, 22:1410–1418, 2009.

[19] E. Papalexakis, C. Faloutsos, and N. Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. *Machine Learning and Knowledge Discovery in Databases*, pages 521–536, 2012.

[20] A.K. Smilde, J.A. Westerhuis, and R. Boque. Multiway multiblock component and covariates regression models. *Journal of Chemometrics*, 14(3):301–331, 2000.

[21] G. Tomasi and R. Bro. Parafac and missing values. *Chemometrics and Intelligent Laboratory Systems*, 75(2):163–180, 2005.

[22] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *SIGCOMM Workshop on Social Networks*, 2009.

[23] T. Wilderjans, E. Ceulemans, and I. Van Mechelen. Simultaneous analysis of coupled data blocks differing in size: A comparison of two weighting schemes. *Computational Statistics & Data Analysis*, 53(4):1086–1098, 2009.

[24] Tatsuya Yokota, Andrzej Cichocki, and Yukihiko Yamashita. Linked parafac/cp tensor decomposition and its fast implementation for multi-block tensor analysis. In *Neural Information Processing*, pages 84–91. Springer, 2012.

[25] Q. Zhang, M. Berry, B. Lamb, and T. Samuel. A parallel nonnegative tensor factorization algorithm for mining global climate data. *Computational Science–ICCS 2009*, pages 405–415, 2009.

[26] Vincent W Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In *AAAI*, 2010.